In this manual the screen representation of "Framework" and any reference to it connotes Design Framework II™ software.

Cover painting, "Epic Bloom," by Gregory Deane.

Cadence Trademarks

Access™
Allegro™
Allegro-MCM™
Amadeus™
Analog Artist™
Analog Workbench™
Analyzer™
ASIC Workbench™
AXL™
BitGrade®
Cadence SPICE™
CAEviews™
Communications Manager™
Component Information Workbench™
Compose™
Composer™
Concept™
Confirm™
Construct™
Dantes™
Design Framework™
Design Framework II™
Design Manager™
Design Planner™
DF/Assembly™
DFFab™
DF/Signal Integrity™
DFTestprep™
DF/Thermax™
DF/Viable™
DIVA®
DLM Place & Route System™
DRACULA®
EDGE™
Ensemble™
GDSII™
GED™
HDL Synthesizer™
Hierarchy Manager™
INSIGHT™
Integrator's Toolkit™
LayDe®
Leapfrog™
License Manager™
Logic Workbench™
MLM Place & Route System™
ModuleMaker™
Open HDL Toolkit™
OpenSim Backplane™
Optimizer™
Opus™
PIC Designer™
PowerVHDL™
PRANCE®
Prance-XL™
Preview™
Process Manager™
Profile™
RapidPART™
RapidSIM™
RapidTEST™
Sage™
SCALDsystem™
Simukit™
SKILL™
Smoke Alarm™
Spectre™
SPICE PLUS™
Structure Compiler™
SYMBAD®
Synergy™
SystemPGA™
SystemPLD™
System Workbench™
Tancell™
Tansure™
Test Generator™
TestGrade®
Test-intelligent Design Series™
TestScan®
Test Simulator™
Test Synthesizer™
ThermoSTATS™
Transcribe™
ValidCOMPILER™
ValidFrame™
ValidGED™
ValidPACKAGER™
ValidSIM™
ValidTIME™
VDoc 454™
Verifault-XL®
Verilog®
Verilog-XL™
Veritime™
Veritools™
VHDL Synthesizer™
VHDL-XL™
Virtuoso™
Warp-4™
Warp Grid™
XLProcessor™

Other Trademarks

UNIX is a registered trademark, licensed exclusively by X/Open Company Ltd.
X Window System is a trademark of the Massachusetts Institute of Technology.

P.1.

Introducing Design Framework II

The Design Framework II™ architecture is the foundation of the Cadence design environment. Design Framework II provides a common user interface, a common functionality, and a common database for the software design tools you use.

The Cadence software consists of many application programs for tasks such as layout, compaction, and verification. These applications run on many hardware platforms. Cadence's open architecture also permits integration of your own tools and tools from other vendors. In such a setting, it is important that the tools and data are well integrated. Design Framework II provides that integration.

P.2

About This Manual

This manual assumes that you are familiar with the development and design of integrated circuits. It contains reference information about the Virtuoso™ Layout Editor. You can use the Layout Editor to

- Draw and edit bends, contacts, polygons, paths, rectangles, circles, ellipses, donuts, pins, transmission lines, and tapers in layout designs.
- Place cells into other cells to create hierarchical designs.
- Create special *parameterized cells* (called *pcells*) containing data that you want to modify quickly or that you want to set with SKILL commands. You can modify one or more pcells by changing the value of a parameter. For detailed information on creating parameterized cells, refer to the *Virtuoso Parameterized Cell Reference Manual.*
- Create and edit analog circuits at a device level.

Finding Information in This Manual

The following chart summarizes the topics covered in this manual.

For information about . . .	Read . . .
Basic features of the Layout Editor	Chapter 1, "Introduction"
Working with drawing layers	Chapter 2, "Layer and Selection Window"
Using Layout Editor commands	Chapter 3, "Layout Editor Commands"
Interactive verification commands	Chapter 4, "Verify Menu"
Using the Device-Level Editor (DLE)	Chapter 5, "Device-Level Editor Commands"

P.3

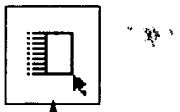
For information about . . .	Read . . .
Using the Microwave Editor	Chapter 6, "Microwave Editor Commands"
Using the Abstract Editor	Chapter 7, "Abstract Editor Commands"
The SKILL functions related to menu commands	Appendix A, "SKILL Cross-Reference Table"
The different ways to start Layout Editor commands	Appendix B, "Starting Layout Editor Commands"
The default keyboard and mouse bindings	Appendix C, "Keyboard and Mouse Bindings"

Conventions

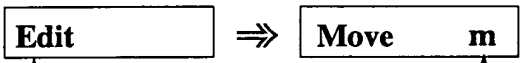
Here are some conventions used to describe menu commands.

Every command description begins with a heading like the one below.

Move



Command icons are shown under the command title.



Boxes and arrows in a sequence show you the order in which you select a command from a menu.

Keyboard bindings are shown next to the command name in the last box.

Each form shows you the system defaults:

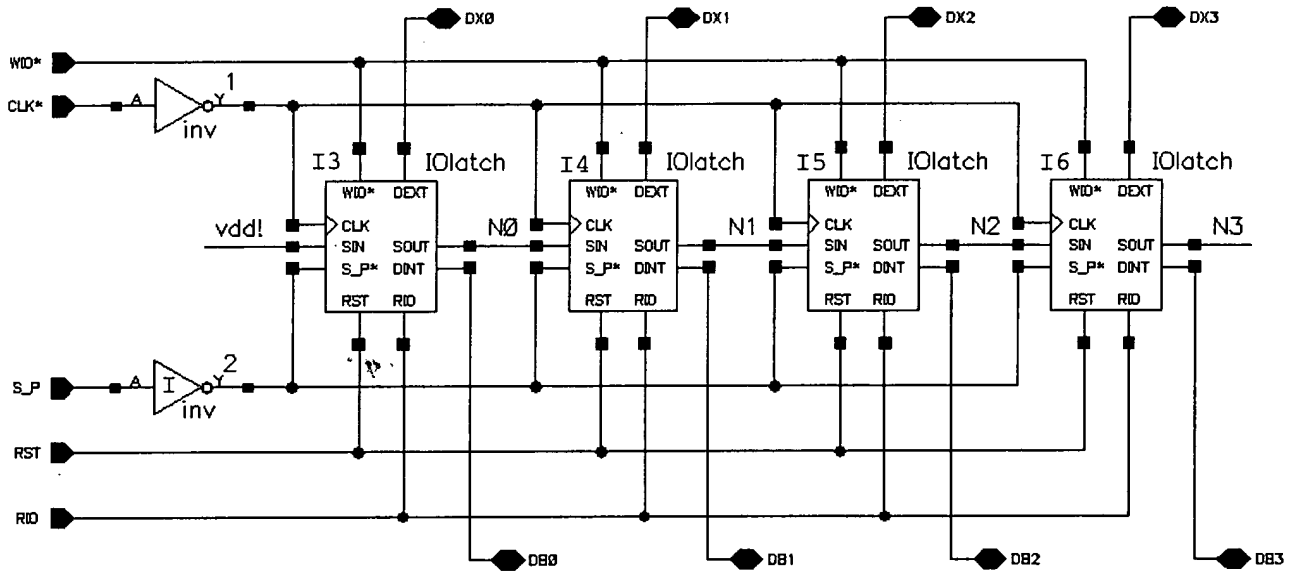
- Filled buttons are the default selections.
- Filled-in values are the default values.

p4

About This Manual

The *Design Entry: Composer User Guide* supports the work of logic and circuit design engineers, including drafters. Physical layout designers and printed circuit board (PCB) designers might find the information useful as background material to support their work.

This guide presents the design entry procedures you can use to develop electronic circuit designs like the following.



IO Latch Schematic

Design entry is only one part of the design process. After you complete your design, refer to the "Other Sources of Information" section of this chapter to learn how to

- Simulate your designs to verify functionality
- Create a layout from your design
- Use a layout to produce a physical chip or printed circuit board (PCB)

p5

Finding Information in This Manual

The following table summarizes the topics covered in this manual.

For Information About . . .	Read . . .
Getting started and setting up your system	Chapter 1, "Getting Started"
Creating a simple design	Chapter 2, "Creating a Design"
Naming objects and connecting them physically	Chapter 3, "Understanding Connectivity"
Creating a symbol that is not in a library	Chapter 4, "Creating a Symbol"
Creating a schematic with several pages or sheets	Chapter 5, "Creating a Multisheet Schematic"
Checking a design for errors	Chapter 6, "Checking a Design"
Modifying a design by copying or moving objects	Chapter 7, "Editing a Design"
Modifying a design by changing the characteristics of an object	Chapter 8, "Updating Properties"
Printing a hard copy of a design	Chapter 9, "Plotting a Design"
Customizing the environment	Chapter 10, "Customizing Design Entry"
Learning terms, definitions, and concepts	Appendix A, "Glossary"

Reader's Input

Please fill out the *Reader's Profile* in the front of this manual. It is important for us to know about your background and the job you do.

Also, please fill out the *Reader's Critique* at the back of this manual to report errors or suggest improvements to us.

p6

The Cadence User Interface

The Cadence software has a graphic user interface based on windows, forms, and menus.

The Operating Environment

The Cadence software runs on workstations manufactured by companies including Digital Equipment Corp., Hewlett-Packard, IBM Corp., and Sun Microsystems, Inc.

Cadence software runs under the manufacturer's variant of the UNIX® operating system. The software runs in graphics mode and nongraphics mode. You must run the software in graphics mode to use the interface described in this manual.

A *windowing system* lets you manage several tasks at a time from separate areas of the screen called windows. Cadence software runs under the X Window System™ (sometimes simply called "X"), developed at the Massachusetts Institute of Technology.

A *window manager* controls the size, placement, and behavior of windows. Cadence software follows the standards established by the Open Software Foundation in its Motif® Window Manager. This manual assumes that you use the Motif Window Manager. The screen illustrations match the appearance of the screens under Motif.

Cadence software accepts input from the keyboard and from a standard three-button mouse. The software sends output to the screen, to disk files, to printers, and to plotting devices.

See the *Design Framework II User Guide* for information on starting the Cadence software and working with the windowing system and window manager. See the *Cadence Configuration Guide* for high-level system setup information.

P.7

Types of Design Framework II Windows

Most of your work with the Cadence software takes place in windows. In Design Framework II, there are five major kinds of windows.

- The *Command Interpreter Window (CIW)* controls your design session. See Chapters 2 through 7 for more information about the CIW.
- *Flowchart Browser windows* display flowcharts that graphically represent steps in the design process. See Chapters 8 and 9.
- *Library Browser windows* display a tree structure containing the names of designs that you can choose for work. See Chapters 10 through 19.
- *Design windows* display and edit your designs. You can have several design windows open at once, each running the same or a different design tool. See the manual for your application for information on design windows created by the application.
- *Text windows* display text files. You can have several text windows open at once. Some text windows dynamically update the display to show new data from a running process. Others display text and run a text editor. See Chapters 20 and 21.

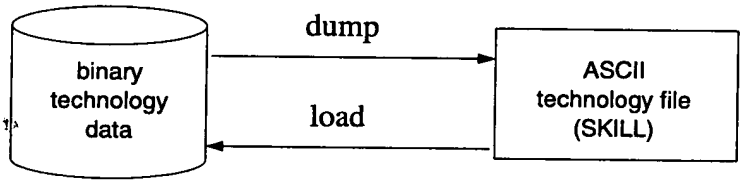
There are other, specialized windows that display the results of specialized processes. For example, a waveform window displays the results of a circuit simulation.

p.8

Technology File Overview

Each cell library is associated with technology data. The technology data defines layers, views, design rules, and symbolic devices for the library.

Technology data is stored in binary format in the cell library. You can use the Technology File menu commands to dump the binary technology data to an ASCII technology file. The term "dump" refers to the process of converting the binary technology data to an ASCII technology file (see figure) that you can edit in a text editor. In the ASCII file, a sequence of SKILL™ functions represents the technology data. Each SKILL function defines an attribute of the technology file data in an easily readable tabular format. After you edit the technology file in a text editor, you can modify the technology data stored in a library by loading the edited file.



You can dump and edit the technology data for a particular application. See the *Design Framework II SKILL Functions Reference Manual* and Appendix A of this manual for the syntax of SKILL functions discussed in this chapter, including the functions with the *tfc* prefix and the database access *db* functions. The SKILL functions associated with the technology file begin with the *tfc* or *tc* prefix.

p. 9

Technology File Contents

The technology file defines all the physical information required for a design. This includes information about

- layers
- colors, line styles, and stipple patterns
- display and plotter devices
- single-layer and two-layer properties
- views and view properties
- physical design rules used for compaction, symbolic checking, and interactive verification
- devices, such as transistors, contacts, and pins; some commonly used devices are provided, and you can add your own device definitions

A technology file is an ASCII text file consisting of a list of SKILL functions. The technology data is compiled when you create a new library or when you load a technology file into a library using the Technology File commands. You can view the contents of this file to see which SKILL functions are compiled into the default binary file.

Important

It is best to specify an existing technology file when you create a library. If you do not specify a technology file name when you create a library, the system automatically loads the layer information from the *techfile.def* file in the install_dir/*etc* (where install_dir is the top directory in which you keep Cadence software files) directory. *techfile.def* contains only the minimum amount of technology data necessary to create a new library. You must specify additional technology data, such as the data defined in *mpu.tf,* to do design work.

The technology file that you use for a new library can be based on *default.tf* or *mpu.tf.*These technology files are in install_dir/*samples/techfile*. Make a copy of *default.tf* or *mpu.tf* and edit the copy to conform to your design requirements. *mpu.tf* contains sample technology data for a two-layer metal CMOS process.

P.10

Layer Properties

The following table lists the layer properties and the applications that use them.

Layer properties and the applications that use them

Property name	Application
areaCap	Block Ensemble, Cell Ensemble
conducting	Cell Ensemble, Layout Editor
contactLimit	Cell Ensemble
contributesToChangedLayer?	Layout Editor
currentDensity	Block Ensemble, Layout Editor
defaultWidth	Layout Editor
edgeCapacitance	Cell Ensemble
eqPinLimit	Cell Ensemble
exit	Cell Ensemble
function	Block Ensemble, Cell Ensemble, Compactor, Layout Editor
horizontalJogLength	Cell Ensemble
lasType	Layout Synthesizer
maskNumber	Cell Ensemble, Block Ensemble, Compactor, Layout Editor
minEnclosure (two-layer property)	Block Ensemble, Cell Ensemble Compactor
minOverlap (two-layer property)	Block Ensemble, Compactor

Layer properties and the applications that use them, continued

Property name	Application
minNotch	Block Ensemble, Layout Editor
minSpacing (two-layer property)	Cell Ensemble, Compactor
minWidth	Cell Ensemble, Block Ensemble, Layout Editor, Compactor
parallelWireCap	Block Ensemble, Cell Ensemble
perimeterCap	Block Ensemble, Cell Ensemble
prRoutingDir	Cell Ensemble, Block Ensemble, Gate Ensemble, Cell3 Ensemble
routingGrid	Cell Ensemble, Layout Editor
routingOffset	Cell3 Ensemble
routingPitch	Cell3 Ensemble
selfEnclosure	Block Ensemble, Compactor
sheetRes	Block Ensemble, Cell Ensemble
streamDatatypeNumber	Stream In/Out Translator
streamLayerNumber	Stream In/Out Translator
streamTranslateLayer	Stream In/Out Translator
verticalJogLength	Cell Ensemble

P. 12

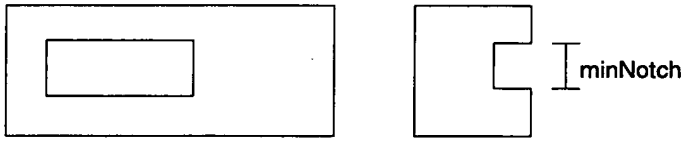
Single-Layer Properties

Single-layer properties set application-specific rules for all objects on a specific layer.

- *areaCap* sets the loading capacitance of the layer in farads per square meter (the same as picofarads per square micron). This property is used for verification. Its value is a floating-point number.
- *conducting* sets the layer routing order. Layers with lower *conducting* values are routed first. A *metal1* layer with a *conducting* value of 1 is routed before a *metal2* layer with a *conducting* value of 2. This property is used for routing. Its value is an integer.
- *contactLimit* limits the number of contacts on this layer. This property is used for routing. Its value is an integer.
- *contributesToChangedLayer?* controls incremental design rule checking, and can be *t* or *nil*. Layers to be checked should have this property set to *t*; other layers should have this property set to *nil*.
- *currentDensity* specifies the amount of current that the default wire width can carry. Its value is a floating-point number.
- *defaultWidth* is the default width used when you create a path in the Layout Editor. Its value is a floating-point number.
- *edgeCapacitance* sets the capacitance between parallel objects. Its value is a floating-point number.
- *eqPinLimit* determines whether the global router can route to electrically equivalent pins. This property is used for routing. Its value is an integer.
- *exit* determines whether wires can route to the channel exit. This property is used for routing. Its value is an integer.
- *function* gives the use for this layer. The functions are *cellBoundary*, *conduction*, *diffusion*, *hardFence*, *implant*, *softFence*, *via*, or *well*. Both the advanced layout editor and the compactor require you to assign layer functions.

P. 13

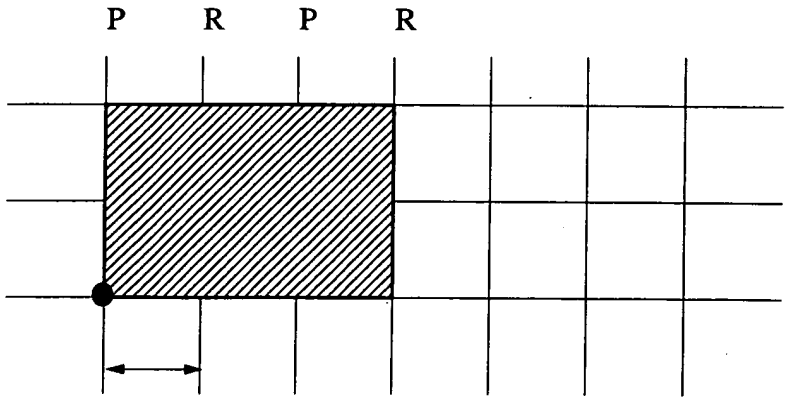
- *horizontalJogLength* limits horizontal routing on this layer. This property is used for routing. Its value is a floating-point number.
- *lasType* assigns the layer type for the Layout Synthesizer. The valid types are *ndiffLayer, pdiffLayer, polyLayer, met1Layer, met2Layer, pwellLayer,* and *nwellLayer*.
- *maskNumber* assigns a number giving the sequence of the layer beginning from the substrate. The mask number also identifies functionally equivalent layers for the mask. For example, you can assign *digital_metal* and *analog_metal* the same mask number. This property is used for routing. Its value is an integer.
- *minNotch* sets the minimum distance between the outside facing edges of a notch drawn in an object. Its value is a floating-point number



- *minWidth* sets the minimum width of the layer. Its value is a positive floating-point number.
- *parallelCap* determines the fringe capacitance for this layer. This property is used for verification. Its value is a floating-point number.
- *perimeterCap* determines the perimeter capacitance for this layer. This property is used for verification. Its value is a floating-point number.
- *prRoutingDir* sets the routing direction for a layer. Its values are *"horizontal"* or *"vertical"*.
- *routingGrid* sets the routing grid multiple. Each wire must be drawn on a multiple of the routing grid value. Its value is a floating-point number.

p. 14

- *routingOffset* is the distance between the placement grid and the routing grid when there is a routing grid between two placement grids.



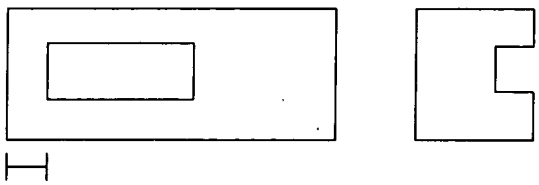
The lines under "P" represent placement grids, and the lines under "R" represent the routing grids. The dot represents the cell origin. The routing offset is the distance between the placement grid and the routing grid.

This layer property applies only to Cell3 Ensemble™. Its value is a floating-point number.

- *routingPitch* defines the minimum spacing between two regular geometries on different nets. Its value is the edge-to-edge separation, both orthogonal and diagonal. *routingPitch* applies to line-to-line, line-to-via, and via-to-via distances. Its value is a floating-point number.

P. 15

- *selfEnclosure* defines the minimum distance between two shapes on this layer if one encloses the other. In most cases, this property is needed only for a layer with the function *cellBoundary*. Its value is a floating-point number.



selfEnclosure

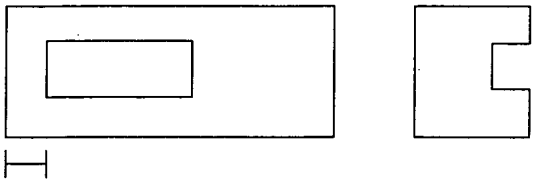
- *sheetRes* sets the resistivity of the layer in ohms per square. Its value is a floating-point number.
- *streamDatatypeNumber* assigns this layer a data type number for GDSII stream out or stream in. Its value is an integer.
- *streamLayerNumber* can be a number from 0 to 63, and assigns this layer a number for GDSII stream out or stream in. Its value is an integer.
- *streamTranslateLayer?* controls whether this layer is translated in GDSII stream out, which can be *t* or *nil*.
- *verticalJogLength* limits vertical routing on this layer. This property is used for routing. Its value is a floating-point number.

p. 16

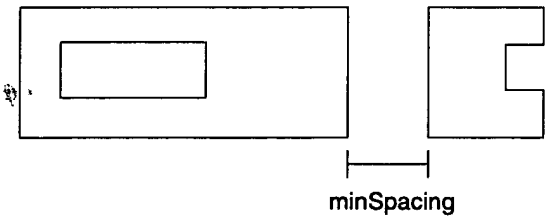
Two-Layer Properties

Two-layer properties set rules between objects on two particular layers. For example, a two-layer property defines the minimum spacing between the n-diffusion and p-diffusion layers.

- *minEnclosure* defines enclosure layers, and optionally specifies the distance by which a shape on layer2 must be enclosed by another shape on layer1. If you want one layer to enclose another, you must set a *minEnclosure*, even if the value is zero. Its value is value a floating-point number.



- *minOverlap* defines the distance by which a shape on layer1 must overlap a shape on layer2. Its value is a floating-point number.
- *minSpacing* sets the minimum external spacing allowed between the two layers. Its value is a floating-point number.



P. 17

What Is a Parameterized Cell?

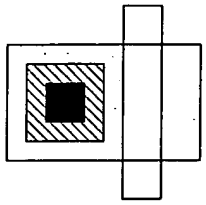
A parameterized cell, or *pcell*, is a graphic, programmable cell that lets you create a customized instance each time you place it. You create this cell just as you create any other layout cell, then you assign parameters to it. The pcell you create is called a *supermaster*. A supermaster is the combination of the graphic layout you draw to make a cell and the parameters you assign to it. After you compile the supermaster, it is stored in the database in the form of a SKILL procedure.

The edits you make to the supermaster appear in the cell instances when you compile the supermaster. You must make all changes to the supermaster. You cannot edit an instance of a pcell.

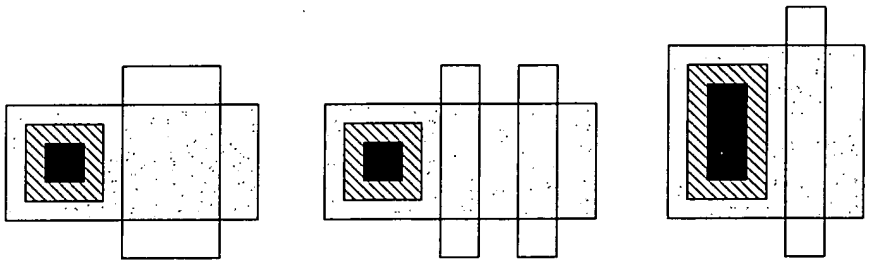
For example, you can create a transistor and assign parameters that control its width, length, and number of gates. When you place instances of the supermaster, you can assign different values to each of these parameters. According to the parameter values, each instance varies in size and composition.

You can create many variations of a transistor using the pcell supermaster. You can use the following supermaster to create an instance with a stretched gate width, an instance with a repeated gate, and an instance with a stretched contact and gate length.

P.18

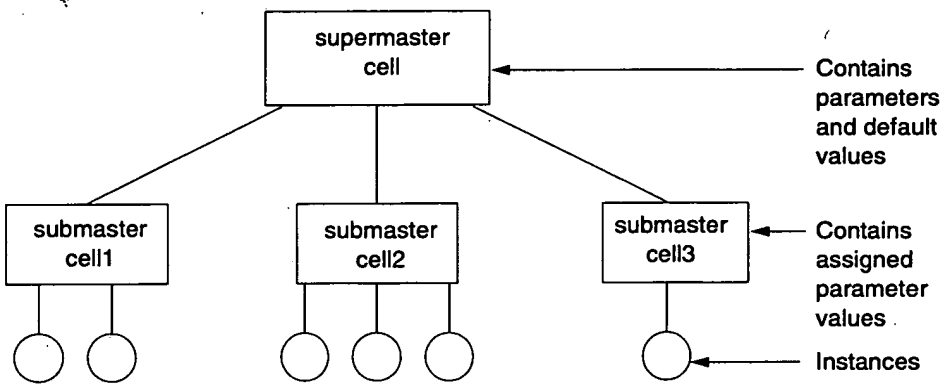


Original transistor (supermaster)



Instance 1 Instance 2 Instance 3

When you place an instance of a pcell, the system creates a submaster with the parameter values you specify. Submaster cellviews are temporary objects that are never stored to disk. They are stored in virtual memory. If you place several instances with the same parameter values, they share the same submaster cellview.



P. 19

When you place an instance of a pcell, you can accept the defaults or change any of the values for the assigned parameters. Using the default values creates an instance identical to the supermaster cellview.

The editor treats instances of pcells just like it treats instances of all other predefined cells. An instance of a pcell is almost identical to an instance of any other cellview to all applications that use the Design Framework II database.

Advantages in Using a pcell

You use pcells to

- Speed up entering layout data by eliminating the need to create duplicate versions of the same functional part
- Save disk space by creating a library of cells for similar parts that are all linked to the same source
- Eliminate errors that can result in maintaining multiple versions of the same cell
- Eliminate the need to explode levels of hierarchy when you want to change a small detail of a design

P.20